IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD

Publication number: JP2006518056 (T)
Publication date: 2006-08-03
Inventor(s):
Applicant(s):
Classification:
- international: ***G06F11/00; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78;*** **G06F11/00; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/76**
- European: G06F9/38S4
Application number: JP20050502223T 20031031
Priority number(s): US20020422503P 20021031; US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; WO2003US34556 20031031

Also published as:
WO2004042560 (A2)
WO2004042560 (A3)
WO2004042574 (A2)
WO2004042574 (A3)
WO2004042569 (A2)
more >>

Abstract not available for JP 2006518056 (T)
Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations.; By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2006-518056
(P2006-518056A)
(43)公表日　平成18年8月3日(2006.8.3)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 11/00	(2006.01)	G06F	9/06	630A	5B076
					5B176

審査請求 未請求　予備審査請求 未請求　（全 27 頁）

(21)出願番号	特願2005-502223 (P2005-502223)
(86)(22)出願日	平成15年10月31日 (2003.10.31)
(85)翻訳文提出日	平成17年6月16日 (2005.6.16)
(86)国際出願番号	PCT/US2003/034556
(87)国際公開番号	WO2004/042569
(87)国際公開日	平成16年5月21日 (2004.5.21)
(31)優先権主張番号	60/422,503
(32)優先日	平成14年10月31日 (2002.10.31)
(33)優先権主張国	米国(US)
(31)優先権主張番号	10/683,929
(32)優先日	平成15年10月9日 (2003.10.9)
(33)優先権主張国	米国(US)
(31)優先権主張番号	10/683,932
(32)優先日	平成15年10月9日 (2003.10.9)
(33)優先権主張国	米国(US)

(71)出願人　504242618
ロッキード　マーティン　コーポレーション
アメリカ合衆国　メリーランド州　２０８１７－１８０３　ベセスダ　ロックレッジ　ドライブ　６８０１
(74)代理人　100083932
弁理士　廣江　武典
(74)代理人　100129688
弁理士　武川　隆宣
(74)代理人　100129676
弁理士　▲高▼荒　新一
(74)代理人　100130074
弁理士　中村　繁元

最終頁に続く

(54)【発明の名称】プログラマブル回路、関連計算マシン、並びに、方法

(57)【要約】
プログラマブル回路は、外部ソースからコンフィギュレーション・データを受信し、ファームウェアをメモリに記憶してから、そのメモリからそのファームウェアをダウンロードする。そうしたプログラマブル回路は、計算マシン等のシステムにプログラマブル回路のコンフィギュレーションを変更させて、コンフィギュレーション・メモリをマニュアルで再プログラムする必要性をなくしている。例えば、もしプログラマブル回路がパイプライン加速器の部分であるＦＰＧＡであれば、その加速器と結合されたプロセッサはＦＰＧＡのコンフィギュレーションを変更できる。より詳細には、プロセッサはコンフィギュレーション・レジストリから変更されたコンフィギュレーションを表すファームウェアを検索し、そのファームウェアをＦＰＧＡに送信し、次いで該ＦＰＧＡがそのファームウェアを電気的に消去可能でプログラム可能な読み取り専用メモリ（ＥＥＰＲＯＭ）等のメモリに記憶する。次に、ＦＰＧＡはメモリからそのコンフィギュレーション・レジスタにそのファームウェアをダウンロードし、よってそれ自体を変更されたコンフィギュレーションを有するように再構成する。

【特許請求の範囲】
【請求項１】
プログラマブル回路であって、
外部ソースからコンフィギュレーションを表すファームウェアを受信し、
前記ファームウェアをメモリに記憶し、
前記メモリから前記ファームウェアをダウンロードするように動作できるプログラマブル回路。
【請求項２】
前記ファームウェアの前記メモリからのダウンロード後に前記コンフィギュレーションで更に動作できる、請求項１に記載のプログラマブル回路。
【請求項３】
前記メモリが不揮発性メモリを含む、請求項１に記載のプログラマブル回路。
【請求項４】
前記メモリが外部メモリを含む、請求項１に記載のプログラマブル回路。
【請求項５】
プログラマブル回路であって、
メモリから第１コンフィギュレーションを表す第１ファームウェアをダウンロードし、
前記第１コンフィギュレーションで動作し、
前記メモリから第２コンフィギュレーションを表す第２ファームウェアをダウンロードし、
前記第２コンフィギュレーションで動作できるプログラマブル回路。
【請求項６】
前記プログラマブル回路が、
前記第１コンフィギュレーションで動作している一方で外部ソースから前記第２ファームウェアを受信し、
前記第１コンフィギュレーションで動作している一方で前記メモリに前記第２ファームウェアを記憶するように更に動作できる、請求項５に記載のプログラマブル回路。
【請求項７】
プログラマブル回路ユニットであって、
メモリと、
前記メモリと結合されたプログラマブル回路と、を含み、
前記プログラマブル回路が、
前記プログラマブル回路のコンフィギュレーションを表すファームウェアを外部ソースから受信し、
前記ファームウェアを前記メモリに記憶し、
前記メモリから前記ファームウェアをダウンロードするように動作できるプログラマブル回路と、
を含むことから成るプログラマブル回路ユニット。
【請求項８】
前記メモリが電気的に消去可能でプログラム可能な読み取り専用メモリを含む、請求項７に記載のプログラマブル回路ユニット。
【請求項９】
前記プログラマブル回路がフィールド・プログラマブル・ゲート・アレイを含む、請求項７に記載のプログラマブル回路ユニット。
【請求項１０】
プログラマブル回路ユニットであって、
それぞれ第１及び第２のコンフィギュレーションを表す第１及び第２のファームウェアを記憶するように動作できるメモリと、
前記メモリと結合されたプログラマブル回路と、を含み、
前記プログラマブル回路が、

前記メモリから前記第１ファームウェアをダウンロードし、
前記第１コンフィギュレーションで動作し、
前記メモリから第２ファームウェアをダウンロードし、
前記第２コンフィギュレーションで動作するように動作できることから成るプログラマブル回路ユニット。

【請求項１１】
前記プログラマブル回路が、
前記第１コンフィギュレーションで動作している一方で外部ソースから前記第２ファームウェアを受信し、
前記第１コンフィギュレーションで動作している一方で前記メモリに前記第２ファームウェアを記憶するように更に動作できる、請求項１０に記載のプログラマブル回路ユニット。

【請求項１２】
前記プログラマブル回路が、前記第１コンフィギュレーションで動作している一方で前記第２ファームウェアをロードするように動作できる、請求項１０に記載のプログラマブル回路ユニット。

【請求項１３】
プログラマブル回路ユニットであって、
それぞれ、第１、第２、第３、並びに、第４のコンフィギュレーションを表す第１、第２、第３、並びに、第４のファームウェアを記憶するように動作できるメモリと、
前記メモリと結合された第１プログラマブル回路と、を含み、
前記第１プログラマブル回路が、
前記メモリから前記第１ファームウェアをダウンロードし、
前記第１コンフィギュレーションで動作し、
前記メモリから第２ファームウェアをダウンロードし、
前記第２コンフィギュレーションで動作するように動作でき、
前記メモリと結合されると共に前記第１プログラマブル回路と結合された第２プログラマブル回路を更に含み、
前記第２プログラマブル回路が、
前記メモリから前記第３ファームウェアをダウンロードし、
前記第３コンフィギュレーションで動作し、
前記メモリから第４ファームウェアをダウンロードし、
前記第４コンフィギュレーションで動作するように動作できることから成るプログラマブル回路ユニット。

【請求項１４】
前記第１プログラマブル回路が、
前記第１コンフィギュレーションで動作している一方で外部ソースから前記第２及び第４のファームウェアを受信し、
前記第１コンフィギュレーションで動作している一方で前記メモリに前記第２及び第４のファームウェアを記憶するように動作できる、請求項１３に記載のプログラマブル回路ユニット。

【請求項１５】
前記第１及び第２のプログラマブル回路が各フィールド・プログラマブル・ゲート・アレイを含む、請求項１３に記載のプログラマブル回路ユニット。

【請求項１６】
計算マシンであって、
プロセッサと、
前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、
前記プログラマブル回路ユニットが、
メモリと、

前記メモリと結合されたプログラマブル回路と、を含み、
前記プログラマブル回路が、
前記プログラマブル回路のコンフィギュレーションを表すファームウェアを前記プロセッサから受信し、
前記メモリに前記ファームウェアを記憶し、
前記プロセッサに応じて前記メモリから前記ファームウェアをダウンロードするように動作できることから成る計算マシン。

【請求項１７】
前記プロセッサが、
前記ファームウェアを前記プログラマブル回路に送信する前、そのファームウェアが前記メモリに既に記憶されているかを決定し、
前記ファームウェアが前記メモリに未だ記憶されていない場合だけ、前記プログラマブル回路に前記ファームウェアを送信するように動作できる、請求項１６に記載の計算マシン。

【請求項１８】
前記プロセッサと結合されると共に、前記ファームウェアを記憶するように且つ前記ファームウェアが前記プログラマブル回路用の所望コンフィギュレーションを表していることを示すように動作できるコンフィギュレーション・レジストリと、
前記プロセッサが前記ファームウェアを前記コンフィギュレーション・レジストリから前記プログラマブル回路にダウンロードするように動作できることと、
を更に含む、請求項１６に記載の計算マシン。

【請求項１９】
前記プログラマブル回路ユニットがパイプライン・ユニットを含み、
前記プログラマブル回路がデータに対して動作するように動作できるハードウェアに組み込まれたパイプラインを含む、請求項１６に記載の計算マシン。

【請求項２０】
計算マシンであって、
プロセッサと、
前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、
前記プログラマブル回路ユニットが、
それぞれ第１及び第２のコンフィギュレーションを表す第１及び第２のファームウェアを記憶するように動作できるメモリと、
プログラマブル回路と、を含み、
前記プログラマブル回路が、
前記メモリから前記第１ファームウェアをダウンロードし、
前記第１コンフィギュレーションで動作し、
前記プロセッサに応じて前記メモリから前記第２ファームウェアをダウンロードし、
前記第２コンフィギュレーションで動作するように、
動作できることから成る計算マシン。

【請求項２１】
前記プロセッサが第１試験ポートを含み、
前記プログラマブル回路ユニットが前記第１試験ポートと結合された第２試験ポートを含み、
前記プロセッサが前記第１及び第２の試験ポートを介して前記第１ファームウェアをメモリにロードするように動作できる、請求項２０に記載の計算マシン。

【請求項２２】
前記プロセッサが第１試験ポートを含み、
前記プログラマブル回路ユニットが前記第１試験ポートと結合された第２試験ポートを含み、

前記第１コンフィギュレーションで動作している一方で、前記プログラマブル回路が自己試験を実行するように且つ自己試験データを前記第１及び第２の試験ポートを介して前記プロセッサに提供するように動作でき、
前記プロセッサが、前記自己試験データが前記自己試験の所定結果を示す場合だけ、前記プログラマブル回路に前記メモリから前記第２ファームウェアをダウンロードさせるように動作できる、請求項２０に記載の計算マシン。
【請求項２３】
前記プロセッサが前記第２ファームウェアを前記プログラマブル回路に送信するように動作でき、
前記第１コンフィギュレーションで動作している一方で、前記プログラマブル回路が前記プロセッサに応じて前記第２ファームウェアを前記メモリにロードするように動作できる、請求項２０に記載の計算マシン。
【請求項２４】
計算マシンであって、
プロセッサと、
前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、
前記プログラマブル回路ユニットが、
それぞれ第１、第２、第３、並びに、第４のコンフィギュレーションを表す第１、第２、第３、並びに、第４のファームウェアを記憶するように動作できるメモリと、
前記メモリと結合された第１プログラマブル回路であり、
前記メモリから前記第１ファームウェアをダウンロードし、
前記第１コンフィギュレーションで動作し、
前記プロセッサに応じて前記メモリから第２ファームウェアをダウンロードし、
前記第２コンフィギュレーションで動作するように、
動作できることから成る第１プログラマブル回路と、
前記メモリと結合されると共に前記第１プログラマブル回路と結合された第２プログラマブル回路であり、
前記メモリから前記第３ファームウェアをダウンロードし、
前記第３コンフィギュレーションで動作し、
前記プロセッサに応じて前記メモリから第４ファームウェアをダウンロードし、
前記第４コンフィギュレーションで動作するように、
動作できる第２プログラマブル回路と、
を含む、計算マシン。
【請求項２５】
前記プロセッサが第１試験ポートを含み、
前記プログラマブル回路ユニットが前記第１試験ポートと結合された第２試験ポートを含み、
前記プロセッサが前記第１及び第２の試験ポートを介して前記第１及び第２のファームウェアをメモリにロードするように動作できる、請求項２４に記載の計算マシン。
【請求項２６】
前記プロセッサが第１試験ポートを含み、
前記プログラマブル回路ユニットが前記第１試験ポートと結合された第２試験ポートを含み、
前記第１コンフィギュレーションで動作している一方で、前記第１プログラマブル回路が第１自己試験を実行するように且つ第１自己試験データを前記第１及び第２の試験ポートを介して前記プロセッサに提供するように動作でき、
前記第３コンフィギュレーションで動作している一方で、前記第２プログラマブル回路が第２自己試験を実行するように且つ第２自己試験データを前記第１及び第２の試験ポートを介して前記プロセッサに提供するように動作でき、
前記プロセッサが、前記第１及び第２のプログラマブル回路に、前記第１及び第２の自

己試験データが前記第１及び第２の自己試験の各所定結果を示す場合だけ、前記メモリから第２及び第４のファームウェアをそれぞれロードさせるように動作できる、請求項２４に記載の計算マシン。

【請求項２７】

前記プロセッサが前記第２及び第４のファームウェアを前記第１プログラマブル回路に送信するように動作でき、

前記第１コンフィギュレーションで動作している一方で、前記第１プログラマブル回路が前記プロセッサに応じて前記第２及び第４のファームウェアを前記メモリにロードするように動作できる、請求項２４に記載の計算マシン。

【請求項２８】

前記メモリが、

前記第１プログラマブル回路と結合され且つ前記第１及び第２のファームウェアを記憶するように動作できる第１メモリ区分と、

前記第１及び第２のプログラマブル回路と結合され且つ前記第３及び第４のファームウェアを記憶するように動作できる第２メモリ区分と、

を含む、請求項２４に記載の計算マシン。

【請求項２９】

前記第１及び第２のメモリ区分が第１及び第２の集積回路上にそれぞれ配置されている、請求項２８に記載の計算マシン。

【請求項３０】

方法であって、

プログラマブル回路にその回路のコンフィギュレーションを表すファームウェアを提供し、

前記プログラマブル回路によって前記コンフィギュレーション・データをメモリに記憶し、

前記メモリから前記プログラマブル回路に前記コンフィギュレーション・データをダウンロードすることを含む方法。

【請求項３１】

前記メモリから前記コンフィギュレーション・データをダウンロードした後、前記コンフィギュレーションで動作すること、

を更に含む、請求項３０に記載の方法。

【請求項３２】

方法であって、

第１コンフィギュレーションを表す第１ファームウェアをプログラマブル回路にダウンロードし、

前記プログラマブル回路を前記第１コンフィギュレーションで動作し、

第２コンフィギュレーションを表す第２ファームウェアを前記プログラマブル回路にダウンロードし、

前記第２ファームウェアをダウンロードした後、前記プログラマブル回路を前記第２コンフィギュレーションで動作することを含む方法。

【請求項３３】

前記第２ファームウェアをダウンロードすることが、

前記第２ファームウェアを前記プログラマブル回路に送信し、

前記プログラマブル回路が前記第１コンフィギュレーションで動作している一方で、該プログラマブル回路によって前記第２ファームウェアをメモリにロードし、

前記メモリから前記プログラマブル回路に前記第２ファームウェアをダウンロードすることを含む、請求項３２に記載の方法。

【請求項３４】

前記第２ファームウェアをダウンロードすることが、

前記第２ファームウェアが前記プログラマブル回路と結合されたメモリに記憶されてい

るかを決定し、
前記第２ファームウェアが前記メモリに記憶されていない場合だけ、前記第２ファームウェアを前記プログラマブル回路に送信し、
前記プログラマブル回路が前記第１コンフィギュレーションで動作している一方で、該プログラマブル回路によって前記第２ファームウェアを前記メモリにロードし、
前記メモリから前記プログラマブル回路に前記ファームウェアをダウンロードすることを含む、請求項３２に記載の方法。
【請求項３５】
前記第１コンフィギュレーションで前記プログラマブル回路を動作することが、該プログラマブル回路を試験することを含み、
前記第２ファームウェアをダウンロードすることが前記プログラマブル回路が前記試験を通過した場合だけ前記第２ファームウェアをダウンロードすることを含む、請求項３２に記載の方法。
【請求項３６】
方法であって、
第１及び第２のファームウェアを第１及び第２のプログラマブル回路にそれぞれダウンロードし、
前記第１及び第２のプログラマブル回路を前記第１及び第２のコンフィギュレーションでそれぞれ動作し、
第３及び第４のファームウェアを前記第１及び第２のプログラマブル回路に、前記第１プログラマブル回路を介して、それぞれダウンロードし、
前記第１及び第２のプログラマブル回路を前記第３及び第４のコンフィギュレーションでそれぞれ動作することを含む方法。
【請求項３７】
前記第１及び第２のファームウェアをダウンロードすることが、前記第１及び第２のファームウェアを前記第１及び第２のプログラマブル回路に、試験ポートを介して、ダウンロードすることを含む、請求項３６に記載の方法。
【請求項３８】
前記第１及び第２のプログラマブル回路を前記第１及び第２のコンフィギュレーションで動作することが、前記第１及び第２のプログラマブル回路を試験することを含み、
前記第３及び第４のファームウェアを前記第１及び第２のプログラマブル回路にロードすることが、
前記試験が、前記第１プログラマブル回路が所望通りに機能していることを示す場合だけ、前記第３ファームウェアをロードし、
前記試験が、前記第２プログラマブル回路が所望通りに機能していることを示す場合だけ、前記第４ファームウェアをロードすることを含む、請求項３６に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
＜優先権の請求＞
この出願は、下記の特許文献１に対する優先権を請求するものであり、引用することでここに合体させる。
【特許文献１】米国仮出願第６０／４２２，５０３号（２００２年１０月３１日出願）
【０００２】
＜関連出願の相互参照＞
この出願は、「改善された計算アーキテクチャ、関連システム、並びに、方法」と題された下記の特許文献２、「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された下記の特許文献３、「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された下記の特許文献４、「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法

ｊと題された下記の特許文献５と関連し、これら特許文献は全て２００３年１０月９日に出力され、共通の所有者を有し、引用することでここに合体させる。
【特許文献２】米国出願第１０／６８４，１０２号
【特許文献３】米国出願第１０／６８４，０５３号
【特許文献４】米国出願第１０／６８３，９２９号
【特許文献５】米国出願第１０／６８３，９３２号
【背景技術】
【０００３】
比較的大量のデータを比較的短い期間で処理する通常の計算アーキテクチャは、処理負担を分担する多数の相互接続プロセッサを含む。処理負担を分担することによって、これら多数のプロセッサは、しばしば、所与のクロック周波数で単一プロセッサができるものよりよりも迅速にデータを処理できる。例えば、これらプロセッサの各々はデータの各部分を処理できるか、或は、処理アルゴリズムの各部分を実行できる。
【０００４】
図１は、多数プロセッサ・アーキテクチャを有する従来の計算マシン１０の概略ブロック図である。この計算マシン１０は、マスター・プロセッサ１２と、相互に通信すると共に該マスター・プロセッサとバス１６を介して通信する共同プロセッサ14_1－14_nと、遠隔装置（図１では不図示）から生データを受け取る入力ポート１８と、該遠隔装置に処理データを提供する出力ポート２０とを含む。また、計算マシン１０はマスター・プロセッサ１２に対するメモリ２２と、共同プロセッサ14_1－14_nに対する各メモリ24_1－24_nと、マスター・プロセッサ及び共同プロセッサがバス１６を介して共有するメモリ２６とを含む。メモリ２２はマスター・プロセッサ１２に対するプログラム及び作業メモリの双方の役割を果たし、各メモリ24_1－24_nは各共同メモリ14_1－14_nに対するプログラム及び作業メモリの双方の役割を果たす。共有されたメモリ２６は、マスター・プロセッサ１２及び共同プロセッサ１４がそれらの間でデータを転送すること、ポート１８を介して遠隔装置からデータを転送すること、ポート２０を介して遠隔装置にデータを転送することを可能としている。またマスター・プロセッサ１２及び共同プロセッサ１４は、マシン１０が生データを処理する速度を制御する共通クロック信号を受け取る。
【０００５】
一般に、計算マシン１０は、マスター・プロセッサ１２及び共同プロセッサ１４の間で生データの処理を効果的に分割する。ソナー・アレイ等の遠隔ソース（図１では不図示）は、ポート１８を介して、生データに対する先入れ先出し（ＦＩＦＯ）バッファ（不図示）として作用する共有メモリ２６の１つの区分に生データをロードする。マスター・プロセッサ１２はバス１６を介してメモリ２６から生データを検索して、マスター・プロセッサ及び共同プロセッサ１４はその生データを処理して、バス１６を介して必要に応じてデータをそれらの間に転送する。マスター・プロセッサ１２はその処理データを共有メモリ２６内に規定された別のＦＩＦＯバッファ（不図示）にロードし、遠隔ソースがポート２０を介してこのＦＩＦＯからその処理データを検索する。
【０００６】
演算例において、計算マシン１０は生データに対するｎ＋１個の各演算を順次実行することによって該生データを処理し、これら演算は一体的に高速フーリエ変換（ＦＦＴ）等の処理アルゴリズムを構成する。より詳細には、マシン１０はマスター・プロセッサ１２及び共同プロセッサ１４からのデータ処理パイプラインを形成する。クロック信号の所与の周波数で、そうしたパイプラインはしばしばマシン１０が単一プロセッサのみを有するマシンよりも高速に生データを処理することを可能としている。
【０００７】
メモリ２６内における生データＦＩＦＯ（不図示）からの生データ検索後、マスター・プロセッサ１２はその生データに対して三角関数等の第１番演算を実行する。この演算は第１番結果を生み出し、それをプロセッサ１２がメモリ２６内に規定された第１番結果ＦＩＦＯ（不図示）に記憶する。典型的には、プロセッサ１２はメモリ２２内に記憶された

プログラムを実行し、そのプログラムの制御の下で上述した動作を実行する。プロセッサ１２はメモリ２２を作業メモリとしても使用し得て、当該プロセッサが第１番演算の中間期間に生成するデータを一時的に記憶する。

【０００８】

次に、メモリ２６内における第１番結果ＦＩＦＯ（不図示）からの第１番結果検索後、共同プロセッサ14_1はその第１番結果に対して対数関数等の第２番演算を実行する。この第２番演算は第２番結果を生み出し、それを共同プロセッサ14_1がメモリ２６内に規定された第２番結果ＦＩＦＯ（不図示）に記憶する。典型的には、共同プロセッサ14_1はメモリ24_1内に記憶されたプログラムを実行し、そのプログラムの制御の下で上述した動作を実行する。共同プロセッサ14_1はメモリ24_1を作業メモリとしても使用し得て、当該共同プロセッサが第２番演算の中間期間に生成するデータを一時的に記憶する。

【０００９】

次に共同プロセッサ24_2－24_nは、共同プロセッサ24_1に対して先に議論されたものと同様に、（第２番結果－第（ｎ－１）番）結果に対して（第３番演算－第ｎ番）演算を順次実行する。

【００１０】

共同プロセッサ24_nによって実行される第ｎ番演算は最終結果、即ち処理データを生み出す。共同プロセッサ24_nはその処理データをメモリ２６内に規定された処理データＦＩＦＯ（不図示）内にロードし、遠隔装置（図１では不図示）がこのＦＩＦＯからその処理データを検索する。

【００１１】

マスター・プロセッサ１２及び共同プロセッサ１４は処理アルゴリズムの種々の演算を同時に実行するので、計算マシン１０は、しばしば、種々の演算を順次実行する単一プロセッサを有する計算マシンよりも生データを高速に処理することができる。詳細には、単一プロセッサは、生データから成る先行集合に対する全（ｎ＋１）個の演算を実行するまで、生データから成る新しい集合を検索できない。しかし、以上に議論したパイプライン技術を用いて、マスター・プロセッサ１２は第１演算だけを実行後に生データから成る新しい集合を検索できる。結果として、所与のクロック周波数でこのパイプライン技術は、単一プロセッサ・マシン（図１では不図示）と比較して約ｎ＋１倍だけマシン１０が生データを処理する速度を増大することができる。

【００１２】

代替的には、計算マシン１０は、生データに対するＦＦＴ等の処理アルゴリズムの（ｎ＋１）例を同時に実行することによって該生データを並列して処理し得る。即ち、もしそのアルゴリズムが先行する例において先に記載されたような（ｎ＋１）個の順次演算を含めば、マスター・プロセッサ１２及び共同プロセッサ１４の各々は生データからそれぞれが成る各集合に対して、順次、全（ｎ＋１）個の演算を実行する。その結果として、所与のクロック周波数で、先のパイプライン技術と同様のこの並列処理技術は、単一プロセッサ・マシン（図１では不図示）と比較して約ｎ＋１倍だけマシン１０が生データを処理する速度を増大することができる。

【００１３】

残念ながら、計算マシン１０は単一プロセッサ・計算マシン（図１では不図示）と比べてより迅速にデータを処理できるが、マシン１０のデータ処理速度はしばしばプロセッサ・クロックの周波数より非常に小さい。詳細には、計算マシン１０のデータ処理速度はマスター・プロセッサ１２及び共同プロセッサ１４がデータ処理するのに必要な時間によって制限される。簡略化のため、この速度制限の例はマスター・プロセッサ１２と連携して議論されているが、この議論は共同プロセッサ１４にも適用されることを理解して頂きたい。先に議論されたように、マスター・プロセッサ１２は所望の方式でデータを操作すべくプロセッサを制御するプログラムを実行する。このプログラムはプロセッサ１２が実行する複数の命令から成るシーケンスを含む。残念ながら、プロセッサ１２は典型的には単一命令を実行するために多数のクロック・サイクルを必要とし、そしてしばしばデータの

単一値を処理すべく多数の命令を実行しなければならない。例えば、プロセッサ１２が第１データ値Ａ（不図示）を第２データ値Ｂ（不図示）で乗算することを仮定する。第１クロック・サイクル中、プロセッサ１２はメモリ２２から乗算命令を検索する。第２及び第３クロック・サイクル中、プロセッサ１２はメモリ２６からＡ及びＢをそれぞれ検索する。第４クロック・サイクル中、プロセッサ１２はＡ及びＢを乗算し、そして第５クロック・サイクル中に結果としての積をメモリ２２或は２６に記憶するか、或は、その結果としての積を遠隔装置（不図示）に提供する。これは最長ケースのシナリオであり、その理由は多くの場合にプロセッサ１２はカウンタの初期化及び閉鎖等のオーバーヘッド・タスクに対して付加的なクロック・サイクルを必要とするからである。それ故に、よくてもプロセッサ１２はＡ及びＢを処理すべく５クロック・サイクルを必要とするか，或は、１データ値当たり平均２．５クロック・サイクルを必要とする。

【００１４】

結果として、計算マシン１０がデータを処理する速度は，しばしば，マスター・プロセッサ１２及び共同プロセッサ１４を駆動するクロックの周波数より非常に低い。例えば、もしプロセッサ１２は１．０ギガヘルツ（ＧＨｚ）でクロックされるが，１データ値当たり平均２．５クロック・サイクルを必要とすれば、効果的なデータ処理速度は（１．０ＧＨｚ）／２．５＝０．４ＧＨｚと同等である。この効果的なデータ処理速度は，しばしば，１秒当たり演算数の単位で特徴付けされる。それ故に，この例において、１．０ＧＨｚのクロック速度で，プロセッサ１２は０．４ギガ演算数／秒（Ｇｏｐｓ）で使用限界が定められる。

【００１５】

図２は、所与クロック周波数で且つしばしば該パイプラインがクロックされる速度と略同一速度で、プロセッサが可能であるよりは高速で典型的にはデータを処理できるハードウェアに組み込まれたデータ・パイプライン３０のブロック線図である。パイプライン３０は、プログラム命令を実行することなく、各データに対する各演算を各々が実行する演算子回路32_1－32_nを含む。即ち、所望の演算は回路３２内に「書き込み」が為されて，それがプログラム命令の必要性なしに自動的にその演算を具現化するように為す。プログラム命令の実行と関連されたオーバーヘッドを減ずることによって，パイプライン３０は所与のクロック周波数でプロセッサが可能であるよりは単位秒当たりより多くの演算を典型的には実行する。

【００１６】

例えば，パイプライン３０は所与のクロック周波数でプロセッサが可能であるよりは高速で以下の数式１をしばしば解くことができる。

$$Y(x_k) = (5x_k + 3)\,2^{x_k}$$

ここで、x_kは複数の生データ値から成るシーケンスを表す。この例において、演算子回路32_1は$5x_k$を計算する乗算器であり，回路32_2は$5x_k+3$を計算する加算器であり，そして回路32_n（ｎ＝３）は$(5x_k+3)\,2^{x_k}$を計算する乗算器である。

【００１７】

第１クロック・サイクルｋ＝１中、回路32_1はデータ値x_1を受け取って、それを５で乗じて、$5x_1$を生成する。

【００１８】

第２クロック・サイクルｋ＝２中、回路32_2は回路32_1から$5x_1$を受け取って、３を加えて、$5x_1+3$を生成する。またこの第２クロック・サイクル中に回路32_1は$5x_2$を生成する。

【００１９】

第３クロック・サイクルｋ＝３中、回路32_3は回路32_2から$5x_1+3$を受け取って，2^{x_1}で乗じて（効果としては、x_1だけ$5x_1+3$を左シフトする）、第１結果$(5x_1+3)\,2^{x_1}$を生成する。またこの第３クロック・サイクル中に回路32_1は$5x_3$を生成し、回路32_2は$5x_2+3$を生成する。

【００２０】

このようにしてパイプライン３０は、全ての生データ値が処理されるまで、引き続く生データ値x_6の処理を続行する。

【００２１】

結果として、生データ値x_1の受け取り後の２つのクロック・サイクルの遅延、即ち、この遅延はパイプライン３０の待ち時間としばしば呼称され、パイプラインは結果（５x_1＋３）2^{x_1}を生成し、その後、１つの結果を生成する、即ち各クロック・サイクル毎に（５x_2＋３）2^{x_2}、（５x_3＋３）2^{x_3}、・・・、５x_6＋３）2^{x_6}を生成する。

【００２２】

待ち時間を無視して、パイプライン３０はこうしてクロック速度と同等のデータ処理速度を有する。比較して、マスター・プロセッサ１２及び共同プロセッサ１４（図１）が先の例におけるようにクロック速度の０．４倍であるデータ処理速度を有すると仮定すれば、パイプライン３０は、所与のクロック速度で、計算マシン１０（図１）よりも２．５倍高速でデータを処理できる。

【００２３】

更に図２で参照されるように、設計者はフィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）等のプログラマブル・ロジックＩＣ（ＰＬＩＣ）にパイプライン３０を具現化することを選ぶ可能性があり、その理由はＰＬＩＣが特殊用途ＩＣ（ＡＳＩＣ）が為すよりも多くの設計及び変更の柔軟性を許容するからである。ＰＬＩＣ内にハードウェアに組み込まれた接続を構成するため、設計者はＰＬＩＣ内に配置された相互接続構成レジスタを単に所定バイナリー状態に設定する。全てのこうしたバイナリー状態の組み合わせはしばしば「ファームウェア」と呼称される。典型的には、設計者はこのファームウェアをＰＬＩＣと結合された不揮発性メモリ（図２では不図示）内にロードする。ＰＬＩＣを「ターンオン」すると、それはファームウェアをそのメモリから相互接続構成レジスタにダウンロードする。それ故に、ＰＬＩＣの機能を変更すべく、設計者は単にそのファームウェアを変更して、ＰＬＩＣがその変更されたファームウェアを相互接続構成レジスタにダウンロードすることを可能とする。ファームウェアを単に変更することによってＰＬＩＣを変更する能力は、モデル作成段階中や「フィールド内」にパイプライン３０をアップグレードするために特に有用である。

【００２４】

残念ながら、ハードウェアに組み込まれたパイプライン３０は重要な意思決定、特に入れ子意思決定を引き起こすアルゴリズムを実行すべき最良の選択でない可能性がある。プロセッサは、典型的には、入れ子意思決定命令（例えば、「もしＡであれば、Ｂを為し、またもしＣであれば、Ｄを為し、・・・、またｎを為し等々」のように、入れ子条件命令）を、比較する長さの演算命令（例えば、「Ａ＋Ｂ」）を実行できる程に高速に実行できる。しかしパイプライン３０は、比較的単純な決定（例えば、「Ａ＞Ｂ？」）を効率的に為し得るが、典型的にはプロセッサができる程に効率的に入れ子決定（例えば、「もしＡであれば、Ｂを為し、またもしＣであれば、Ｄを為し、・・・またｎを為す」）を為すことができない。この非効率性の１つの理由は、パイプライン３０はほんの僅かなオンボード・メモリしか持たないことがあり、したがって外部作業／プログラム・メモリ（不図示）にアクセスすることを必要とすることがあるからである。そして、こうした入れ子決定を実行すべくパイプライン３０を設計することができるが、必要とされる回路のサイズ及び複雑性はしばしばそうした設計を非現実的に為し、特にアルゴリズムが多数の種々の入れ子決定を含む場合にそうである。

【００２５】

結果として、プロセッサは典型的には重要な意思決定を必要とする用途において使用され、ハードウェアに組み込まれたパイプラインは殆ど意思決定が為されないか或は意思決定されない「ナンバークランチング（数値データ処理）」用途に典型的には限定される。

【００２６】

更には、下記に議論されるように、典型的には、特にパイプライン３０が多数のＰＬＩＣを含む場合、図２のパイプライン３０等のハードウェアに組み込まれたパイプラインを

設計／変更するよりも、図１の計算マシン１０等のプロセッサに基づく計算マシンを設計／変更することが非常に易しい。

【００２７】

プロセッサ及びそれらの周辺機器（例えば、メモリ）等の計算構成要素は、典型的には、プロセッサに基づく計算マシンを形成すべくそれら構成要素の相互接続を補助する工業規格通信インターフェースを含む。

【００２８】

典型的には、規格通信インターフェースは２つの層、即ち、物理層及びサービス層を含む。

【００２９】

物理層は、回路とこの回路のインターフェース及び動作パラメータを形成する対応回路相互接続とを含む。例えば、物理層はそれら構成要素を１つのバスに接続するピンと、それらのピンから受け取ったデータをラッチするバッファと、信号をそれらピンに駆動するドライバと、入力データ信号からデータを回復すると共にそのデータ信号或は外部クロック信号からクロック信号を回復する回路とを含む。動作パラメータは、ピンが受け取るデータ信号の許容可能電圧範囲と、データの書き込み及び読み取りのための信号タイミングと、動作の支援されたモード（例えば、バーストモード、ページモード）とを含む。従来の物理層はトランジスタ-トランジスタ論理（ＴＴＬ）及びＲＡＭＢＵＳを含む。

【００３０】

サービス層は、計算構成要素のデータ転送のためのプロトコルを含む。このプロトコルはデータのフォーマットと、構成要素によるフォーマット済みデータの送受信の方式とを含む。従来の通信プロトコルは、ファイル転送プロトコル（ＦＴＰ）及び伝送制御プロトコル／インターネット・プロトコル（ＴＣＰ／ＩＰ）を含む。

【００３１】

結果として、製造業者やその他は工業規格通信インターフェースを有する計算構成要素を典型的には設定するので、そうした構成要素のインターフェースを典型的には設計できて、それを他の計算構成要素と比較的少ない労力で相互接続することができる。これは、計算マシンの他の部分の設計に設計者自信の時間を殆ど費やすことを可能として、各種構成要素を追加或は除去することによってそのマシンを変更することを可能としている。

【００３２】

工業規格通信インターフェースを支援する計算構成要素を設計することは、設計ライブラリから既存の物理層を用いることによって設計時間を節約することを可能としている。これは、設計者が構成要素を既製の計算構成要素と容易にインターフェースすることを保証するものでもある。

【００３３】

そして、共通した工業規格通信インターフェースを支援する計算構成要素を用いる計算マシンを設計することは、設計者がそれら構成要素を少しの時間及び労力で相互接続することを可能としている。それら構成要素は共通インターフェースを支援するので、設計者はそれらをシステム・バスを介して少しの設計労力で相互接続することができる。そして、その支援されたインターフェースは工業規格であるので、マシンを容易に変更することができる。例えば、システム設計が進化するに伴って種々の構成要素及び周辺機器をマシンに追加することができるか、或は、テクノロジーが進化するに伴って次世代の構成要素を追加／設計することが可能である。更には、構成要素が通常の工業規格サービス層を支援するので、計算マシンのソフトウェアに対応するプロトコルを具現化する既存のソフトウェア・モジュールを組み込むことができる。それ故に、インターフェース設計が本質的には既に整っているので少しの労力で構成要素をインターフェースでき、よって、マシンに所望の機能を実行させるマシンの各種部分（例えばソフトウェア）の設計に集中することができる。

【００３４】

しかし残念ながら、図２のパイプライン３０等のハードウェアに組み込まれたパイプラ

インを形成すべく、使用されるＰＬＩＣ等の各種構成要素に対する既知の仕業規格サービス層が全くない。

【００３５】

結果として、多数のＰＬＩＣを有するパイプラインを設計すべく、多大な時間を費やし、「ゼロから」設計し且つ種々のＰＬＩＣの間の通信インターフェースのサービス層をデバッグする多大な労力を行使する。典型的には、そうしたその場限りのサービス層は種々のＰＬＩＣ間で転送されるデータのパラメータに依存する。同じように、プロセッサとインターフェースするパイプラインを設計すべく、パイプライン及びプロセッサの間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行使する必要がある。

【００３６】

同様に、そうしたパイプラインをＰＬＩＣを該パイプラインに追加することによって変更すべく、典型的には、その追加されたＰＬＩＣと既存のＰＬＩＣとの間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行使する。同じように、プロセッサを追加することによってパイプラインを変更すべく、或は、パイプラインを追加することによって計算マシンを変更すべく、パイプライン及びプロセッサの間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行使しなければならいであろう。

【００３７】

結果として、図１及び図２で参照されるように、多数のＰＬＩＣをインターフェースすることとプロセッサをパイプラインにインターフェースすることとの難しさのため、計算マシンを設計する際に多大な妥協を為すことがしばしば強いられる。例えば、プロセッサに基づく計算マシンでは、ナンバークランチング速度を、複雑な意思決定を為す能力に対する設計／変更の柔軟性と交換することを強いられる。逆に、ハードウェアに組み込まれたパイプラインに基づく計算マシンでは、複雑な意思決定を為す能力と設計／変更の柔軟性を、ナンバークランチング速度と交換することを強いられる。更には、多数のＰＬＩＣをインターフェースすることに関する難しさのため、少数のＰＬＩＣよりも多くのＰＬＩＣを有するパイプラインに基づくマシンを設計することはしばしば実際的ではない。その結果、実際的なパイプラインに基づくマシンはしばしば制限された機能しか有さない。そして、プロセッサをＰＬＩＣとインターフェースすることに関する難しさのため、プロセッサを１つのＰＬＩＣより多くのＰＬＩＣにインターフェースすることは実際的ではない。その結果、プロセッサ及びパイプラインを組み合わせることによって獲得される利益は最少となる。

【発明の開示】

【発明が解決しようとする課題】

【００３８】

それ故に、プロセッサに基づくマシンの意思決定を為す能力を、ハードウェアに組み込まれたパイプラインに基づくマシンのナンバークランチング速度と組み合わせることを可能とする新しい計算アーキテクチャに対する要望が生じてきている。

【課題を解決するための手段】

【００３９】

本発明の実施例に従えば、プログラマブル回路は外部ソースからファームウェアを受信し、そのファームウェアをメモリ内に記憶してから、そのファームウェアをそのメモリからダウンロードする。

【００４０】

そうしたプログラマブル回路は、計算マシン等のシステムにプログラマブル回路のコンフィギュレーションを変更させることを可能とし、よってコンフィギュレーション・メモリをマニュアルで再プログラムする必要性を削減している。例えば、もしプログラマブル回路がパイプライン加速器の部分であるＦＰＧＡであれば、その加速器と結合されたプロセッサはＦＰＧＡのコンフィギュレーションを変更できる。より詳細には、プロセッサは

コンフィギュレーション・レジストリからその変更されたコンフィギュレーションを表すファームウェアを検索して、そのファームウェアをＦＰＧＡに送信し、次いで該ＦＰＧＡがそのファームウェアを、電気的に消去可能でプログラム可能な読み取り専用（ＥＥＰＲＯＭ）等のメモリ内に記憶する。次に、ＦＰＧＡはそのメモリからそのコンフィギュレーション・レジストリにファームウェアをダウンロードし、こうして変更されたコンフィギュレーションを有するようにそれ自体を効果的に再構成する。

【発明を実施するための最良の形態】

【００４１】

図３は、本発明の一実施例に従ったピア-ベクトル・アーキテクチャを有する計算マシン４０の概略ブロック線図である。ホストプロセッサ４２に加えて、ピア-ベクトル・マシン４０はパイプライン加速器４４を含み、それがデータ処理の少なくとも一部を実行して、図１の計算マシン１０における共同プロセッサ１４の列と効果的に置き換わる。それ故に、ホストプロセッサ４２及び加速器４４（又は以下に議論されるようにそのパイプライン・ユニット）はデータ・ベクトルを前後に転送できる「ピア」である。加速器４４がプログラム命令を実行しないので、所与のクロック周波数で共同プロセッサの列ができるものよりも著しく高速にデータに対して数学的に集中的な演算を典型的には実行する。結果として、プロセッサ４２の意思決定能力と加速器４４のナンバークランチング能力とを組み合わせることによって、マシン４０はマシン１０等の従来の計算マシンと同一の能力を有するが、しばしばそれよりもデータをより高速に処理することができる。更には、以下に議論されるように、加速器４４にホストプロセッサ４２の通信インターフェースと互換性がある通信インターフェースを設けることが、特にプロセッサの通信インターフェースが工業規格である場合に、マシン４０の設計及び変更を補助する。そして、加速器４４が１つ或はそれ以上のＰＬＩＣを含む場合、ホストプロセッサ４２は適切なファームウェアをそれらＰＬＩＣに送信することによって加速器内における物理的な相互接続コネクタをハード的に構成できる。ホストプロセッサ４２はピア-ベクトル・マシン４０の初期化中にこの方式で加速器４４を構成できるばかりではなく、以下に議論されると共に先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に議論されるように、ピア-ベクトル・マシンの動作中に加速器を再構成する能力を有し得る。更には、ピア-ベクトル・マシン４０は以下に議論されると共に先行して引用された特許出願におけるような他の長所等をも提供し得る。

【００４２】

更に図３で参照されるように、ホストプロセッサ４２及びパイプライン加速器４４に加えて、ピア-ベクトル・計算マシン４０は、プロセッサ・メモリ４６、インターフェース・メモリ４８、パイプライン・バス５０、１つ或はそれ以上のファームウェア・メモリ５２、任意選択的な生データ入力ポート５４、処理済みデータ出力ポート５８、任意選択的なルータ６１、並びに、試験バス６３を含む。

【００４３】

ホストプロセッサ４２は処理ユニット６２及びメッセージ・ハンドラー６４を含み、プロセッサ・メモリ４６は処理ユニット・メモリ６６及びハンドラー・メモリ６８を含み、そのそれぞれがプロセッサ・ユニット及びメッセージ・ハンドラーに対するプログラム及び作業の両メモリとして役立っている。プロセッサ・メモリ４６は、加速器コンフィギュレーション・レジストリ７０及びメッセージ・コンフィギュレーション・レジストリ７２をも含み、それらが、ホストプロセッサ４２に加速器４４の機能とメッセージ・ハンドラー６４が送信及び受信するメッセージのフォーマットと構成させることをそれぞれ可能とするファームウェア及びコンフィギュレーション・データを記憶する。加速器４４及びメッセージ・ハンドラー６４のコンフィギュレーションは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に更に議論され、加速器４４のコンフィギュレーションも図４乃至図６と連携して以下に更に議論される。

【００４４】

パイプライン加速器４４は少なくとも１つのＰＬＩＣ（図４）上に配置され、プログラム命令を実行することなしに各データを処理するハードウェアに組み込まれたパイプライン７４$_1$－７４$_a$を含む。ファームウェア・メモリ５２は加速器４４に対するファームウェアを記憶する。より詳細には、ファームウェア・メモリ５２は、図４乃至図６と連携して以下に更に議論されるように、加速器４４を構成する複数のＰＬＩＣに対するファームウェアを記憶する。代替的には、加速器４４は少なくとも１つのＡＳＩＣ上に配置され得て、そのＡＳＩＣがひとたび形成されたならば構成不可能である内部相互接続を有し得る。加速器４４がＰＬＩＣを何等含まないこの代替例において、マシン４０はファームウェア・メモリ５２を省略し得る。更には、加速器４４が多数パイプライン７４$_1$－７４$_a$を含んで示されているが、ただ１つのパイプラインを含み得る。加えて、図示されていないが、加速器４４はディジタル信号プロセッサ（ＤＳＰ）等の１つ或はそれ以上のプロセッサを含み得る。更には、図示されていないが、加速器４４はデータ入力ポート及び／或はデータ出力ポートを含み得る。

【００４５】

ピアーベクトル・マシン４０の一般動作は、先行して引用された「改善された計算アーキテクチャ、関連システム、並びに、方法」と題された特許文献２に議論されており、ホストプロセッサ４２の構造及び動作は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に議論されており、パイプライン加速器４４の構造及び動作は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献４及び「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法」と題された特許文献５に議論されている。加速器４４を構成するＰＬＩＣの動作コンフィギュレーションは、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献４や、図４乃至図６と連携して以下に議論されている。

【００４６】

図４乃至図６で参照されるように、加速器４４ＰＬＩＣを「ハード」的に構成するための技術が議論される。上記に暗に触れているように、ＰＬＩＣのハード・コンフィギュレーションはファームウェアによってプログラムされ、ＰＬＩＣの種々の構成要素間での特定の物理的相互接続、即ち、どのようにして一方の論理ブロックが他方の論理ブロックと電気的に接続されているかを示す。これは、既にハード的に構成されたＰＬＩＣのより高レベル・コンフィギュレーションを示す「ソフト」コンフィギュレーションとは対照的である。例えば、ハード的に構成されたＰＬＩＣはバッファを含み得ると共にレジスタをも含み得て、そのレジスタによって対応するソフト・コンフィギュレーション・データを該レジスタにロードすることによって、そのバッファのサイズをソフト的に構成させることができる。加速器４４のソフト・コンフィギュレーションは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３及び「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献４に更に議論されている。

【００４７】

図４は、本発明の実施例に従った図３のパイプライン加速器４４のパイプライン・ユニット７８のブロック線図である。ハードウェアに組み込まれたパイプライン７４$_1$－７４$_a$（図３）はパイプライン・ユニット７８の部分であり、それは、以下に議論されるように、例えば、ハードウェアに組み込まれたパイプラインを制御すると共にそれらにデータを受信、送信、そして記憶させることを可能とする回路を含む。１つのみのパイプライン・ユニット７８が図４に示されているが、加速器４４は、先行して引用された「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法」と題された特許文献５に議論されたように、多数のパイプライン・ユニット（各々がハードウェアに組み込まれたパイプライン７４$_1$－７４$_a$の内の少なくとも幾つかを含む）を含み得る。以下に議論されるように、１つの具現化例において、パイプライン・ユニット７８のハ

ード・コンフィギュレーションはファームウェアでプログラム可能である。これは、ファームウェアを単に変更することによって、パイプライン・ユニット７８の機能を変更させることを可能としている。更には、ホストプロセッサ４２（図３）はピア－ベクトル・マシン４０（図３）の初期化中或は再構成中に変更されたファームウェアをパイプライン・ユニット７８に提供でき、よってその変更ファームウェアをパイプライン・ユニットにマニュアルでロードさせる必要性をなくせる。

【００４８】

パイプライン・ユニット７８は，ＰＬＩＣ或はＡＳＩＣ等のパイプライン回路８０、ファームウェア・メモリ５２（パイプライン回路がＰＬＩＣの場合）、並びに、データ・メモリ８１を含み、それら全てが回路ボード或はカード８３上に配置されることになる。データ・メモリ８１は、先行して引用された「プログラマブル回路、関連計算マシン、並びに、方法」と題された米国特許出願第１０／６８４，０５７号に更に議論され、パイプライン回路８０及びファームウェア・メモリ５２の組み合わせはプログラマブル回路ユニットを形成する。

【００４９】

パイプライン回路８０は通信インターフェース８２を含み、それが，ホストプロセッサ４２（図３）等のピアとデータ・メモリ８１との間、そして、ピアと、通信シェル８４を介したハードウェアに組み込まれたパイプライン７４$_1$－７４$_n$、パイプライン・コントローラ８６、例外マネージャ８８、並びに、コンフィギュレーション・マネージャ９０等の、パイプライン回路の他の構成要素との間でデータを転送する。パイプライン回路８０は工業規格バス・インターフェース９１及びインターフェース８２をインターフェース９１と接続する通信バス９３をも含み得る。代替的には、インターフェース９１の機能は通信インターフェース８２内に含まれ得て、バス９３は省略され得る。ハードウェアに組み込まれたパイプライン７４$_1$－７４$_n$、コントローラ８６、例外マネージャ８８、コンフィギュレーション・マネージャ９０，並びに、バス・インターフェース９１の構造及び動作は，先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献４に議論されている。

【００５０】

通信インターフェース８２はメッセージ・ハンドラー６４（図３）によって認識されるフォーマットでデータを（存在する場合にはバス・インターフェース９１を介して）送受信し、よってピア－ベクトル・マシン４０（図３）の設計及び変更を典型的には補助する。例えば、もしデータ・フォーマットが高速Ｉ／Ｏフォーマット等の工業規格であれば，ホストプロセッサ４２及びパイプライン・ユニット７８の間にカスタムインターフェースを設計する必要がない。更には、パイプライン・ユニット７８に非バス・インターフェースの代わりにパイプライン・バス５０を介してホストプロセッサ４２（図３）等の他のピアと通信させることを可能することによって、パイプライン・ユニットが追加或は除去されるたびにスクラッチから非バス・インターフェースを再設計する代わりに、パイプライン・バスにそれらパイプライン・ユニット（又はそれらを保持する回路カード）を単に接続或は接続解除することによってパイプライン・ユニットの数を変更できる。

【００５１】

パイプライン回路８０がＦＰＧＡ等のＰＬＩＣである場合、通信インターフェース８２はプログラマブル・ポート９４を含み、それが以下に議論されるようにパイプライン回路にホストプロセッサ４２（図３）からファームウェア・メモリ５２にファームウェアをロードさせることを可能としている。例えば、もしファームウェア・メモリ５２がＥＥＰＲＯＭであるならば、プログラム・サイクル中にファームウェア・メモリが必要とするプログラミング信号を、通信インターフェース８２は生成し、ポート９４は配送する。そうしたプログラミング信号を生成する回路は従来的なものであり、よって更に議論されない。

【００５２】

通信インターフェース８２の構造及び動作は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献

４に更に議論されている。

【００５３】

更に図４で参照されるように、パイプライン回路８０は、試験ポート９６、パイプライン回路がＰＬＩＣである場合にハード・コンフィギュレーション・ポート９８をも含む。試験バス６３と結合された試験ポート９６は、ホストプロセッサ４２（図３）に、以下に議論されるように、パイプライン回路８０がピア-ベクトル・マシン４０（図３）の初期化中に実行し得る自己試験の結果をモニタさせることを可能とする。製品には、典型的には、パイプライン回路８０を伴う試験ポート９６を含み、そして典型的には、試験ポートにＪＴＡＧ等の工業規格試験プロトコルと互換性があるインターフェース（不図示）を提供する。ハード・コンフィギュレーション・ポート９８は、以下に議論されるように、パイプライン回路８０にメモリ５２からファームウェアをダウンロードすることによってそれ自体を構成させることを可能とする。試験ポート９６のように、製品には、典型的には、パイプライン回路８０を伴うコンフィギュレーション・ポート９８を含み、そして典型的にはコンフィギュレーション・ポートに工業規格メモリ・インターフェースと、メモリ５２の所定アドレス範囲からファームウェアを順次ダウンロードする状態マシンと（双方とも不図示）を提供する。

【００５４】

先に議論されると共に以下に更に議論されるように、パイプライン回路８０がＰＬＩＣである場合、ファームウェア・メモリ５２はパイプライン回路の１組成はそれ以上の組みのハード・コンフィギュレーションを表すファームウェアを記憶する。このファームウェア・メモリ５２は試験ポート１０４と、プログラミング及びコンフィギュレーション・ポート１０６，１０８とを含む。試験バス６３と結合された試験ポート１０４は、ホストプロセッサ４２（図３）に、以下に議論されるように、ファームウェア・メモリ５２がピア-ベクトル・マシン４０（図３）の初期化中に実行し得る自己試験の結果をモニタさせることを可能とする。また以下に議論されるように、試験ポート１０４はホストプロセッサ４２にファームウェアをメモリ５２にロードさせることを可能とし得る。製品には、典型的には、メモリ５２を伴う試験ポート１０４を含み、そして典型的にはその試験ポートにＪＴＡＧ等の工業規格試験プロトコルと互換性があるインターフェース（不図示）を提供する。プログラミング・バス１１０を介して通信インターフェース８２のプログラミング・ポート９４と結合されたプログラミング・ポート１０６は、以下に議論されるようにパイプライン回路８０にファームウェアをメモリ５２にロードさせることを可能とする。そして、コンフィギュレーション・バス１１２を介してパイプライン回路８０のハード・コンフィギュレーション・ポート９８と結合されたハード・コンフィギュレーション・ポート１０８は、以下に議論されるように、パイプライン回路にメモリ５２からファームウェアをダウンロードさせることを可能とする。典型的には、ファームウェア・メモリ５２はＥＥＰＲＯＭ等の不揮発性メモリであり、電力が欠如している状態でデータを保持する。結果として、ファームウェア・メモリ５２はパイプライン・ユニット７８が電力がダウンした後もそのファームウェアを記憶し続ける。

【００５５】

更に図４に参照されるように、ファームウェア・メモリ５２及びデータ・メモリ８１がパイプライン回路８０の外部であると説明されているが、何れか一方或は両方のメモリはパイプライン回路に組み込まれ得る。メモリ５２がパイプライン回路８０の内部に配置されている場合、設計者はそれに対応してプログラミング及びコンフィギュレーション・バス１１０，１１２の構造を変更する必要があり得る。更には、パイプライン・ユニット７８がコンフィギュレーション・バス１１２から分離したプログラミング・バス１１０を有すると説明されているが、単一バス（不図示）はプログラミング及びコンフィギュレーションの両バスの機能を実行し得る。代替的には、パイプライン・ユニット７８はこの単一バスの多数例を含み得るか、或は、プログラミング及びコンフィギュレーション１１２，１１０の一方或は両方の多数例を含み得る。

【００５６】

図５は、本発明の実施例に従った図４のファームウェア・メモリ５２の論理区画の線図である。

【００５７】

メモリ５２の区分１１４はパイプライン回路８０（図４）の初期コンフィギュレーションを表すファームウェアを記憶する。即ち、パイプライン回路８０にダウンロードされると、このファームウェアはパイプライン回路に初期コンフィギュレーションを所有させる。初期コンフィギュレーションの１つの具現化例において、パイプライン回路８０は図４の通信インターフェース８２（そしてもし必要があれば工業規格バス・インターフェース９１）と、パイプライン回路及びデータ・メモリ８１の自己試験を実行する自己試験回路（不図示）とを含む。パイプライン回路８０は、次いで、試験バス６３或は通信インターフェース８２を介して自己試験の結果をホストプロセッサ４２（図３）に提供できる。また初期コンフィギュレーションは、ホストプロセッサ４２に変更されたファームウェアを、以下に議論されるように、通信インターフェース８２及びプログラミング・バス１１０を介してファームウェア・メモリ５２にロードさせることを可能としている。

【００５８】

メモリ５２の区分１１６_1－１１６_nは、各々、パイプライン回路８０の各動作コンフィギュレーションを表すファームウェアを記憶する。典型的には、パイプライン回路８０は、加速器４４（図３）の初期化の最後に区分１１６_1－１１６_nの内の所定の１つからファームウェアをダウンロードする。以下に議論されるように、パイプライン回路８０は特定の区分１１６_1－１１６_nからファームウェアをダウンロードすべく予めプログラムされ得るか、或は、ホストプロセッサ４２（図３）はパイプライン回路に特定の区分からファームウェアをダウンロードするように命令する。典型的には、ｉ個の動作コンフィギュレーションの各々において、パイプライン回路８０は図４に示される構成要素（例えば、ハードウェアに組み込まれたパイプライン７４_1－７４_n、コントローラ８６）を含む。しかし、これらコンフィギュレーションの各々において、パイプライン回路８０は典型的には異なるように動作する。例えば、通信インターフェース８２は一方のコンフィギュレーションにおいて一方のプロトコルを具現化し、他方のコンフィギュレーションにおいて他方のプロトコルを具現化する。或は、パイプライン７４_1－７４_nは一方のコンフィギュレーションにおいてデータに対して一方の組の演算を実行し得て、他方のコンフィギュレーションにおいてデータに対して他方の組の演算を実行し得る。

【００５９】

任意選択的な区分１１８は、メモリ５２の区分１１６_1－１１６_iに記憶されたファームウェアによってそれぞれ表される動作コンフィギュレーションの記述或は識別表示を記憶する。この記述／識別表示はホストプロセッサ４２（図３）にメモリ５２に記憶されたファームウェアを識別させることを可能とする。

【００６０】

任意選択的な区分１２０はパイプライン・ユニット７８（図４）のプロファイルを記憶する。このプロファイルはパイプライン・ユニット７８のハードウェア・レイアウトを充分に記述して、ホストプロセッサ４２（図３）が、それ自体、パイプライン・ユニット、並びに、ピア－ベクトル・マシン４０（図３）の別のピア（不図示）を相互通信のために適切に構成するように為す。例えばプロファイルは、パイプライン・ユニット７８が具現化できるデータ演算及び通信プロトコル、データメモリ８１のサイズ、区分１１６_1－１１６_iに記憶されたファームウェアによって表される動作コンフィギュレーション（もし区分１１８が省略された場合）、並びに、現行において所望された動作コンフィギュレーションを識別し得る。結果として、ピア－ベクトル・マシン４０の初期化中にプロファイルを読み取ることによって、ホストプロセッサ４２は、パイプライン・ユニット７８と通信すべく、メッセージ・ハンドラー６４（図３）を適切に構成できる。更には、ホストプロセッサ４２はパイプライン回路８０がダウンロードすべきファームウェアの区分１１６_1－１１６_iを選択し得る。或は、もしこのファームウェアの何れも適合しなければ、ホストプロセッサ４２は変更されたファームウェアをメモリ５２にロードし得る。この技術は

「プラグ・アンド・プレイ」技術と類似し、それによってコンピュータはそれ自体を構成できて、新しくインストールされたディスク・ドライブ等の周辺機器と通信する。

【００６１】

代替的には、区分１２０は、ホストプロセッサ４２（図３）に、例えば、加速器コンフィギュレーション・レジストリ７０（図３）に記憶されているテーブルからプロファイルを検索させることを可能とする（しばしば「実行インデックス」と呼称される）プロファイル識別子を記憶し得る。典型的には、実行インデックスは数であり、製品の型番号と非常に似ており、ホストプロセッサ４２は記憶されたプロファイルと照合することができる。

【００６２】

更に別の代替例において、パイプライン・ユニット７８（図４）は「ハードウェアに組み込まれた」形態でプロファイル識別子を記憶し得て、区分１２０のプロファイルを改竄ではなく上書きし得る機会をなくする。例えば、パイプライン・ユニット７８は、ホストプロセッサ４２（図３）が試験バス６３或はパイプライン・バス５０及びパイプライン回路８０（図４）を介して読み取ることができるハードウェアに組み込まれた「レジスタ」に、そのプロファイル識別子を記憶し得る。このレジスタは、電気機械式スイッチ、ジャンパー、或は、半田付け接続部（不図示）等で形成され得る。

【００６３】

更に図５で参照されるように、ファームウェア・メモリ５２の任意選択的な区分１２２は、ファームウェア・メモリ５２が加速器４４の初期化中に実行する自己試験ルーチン等のその他のデータを記憶し得る。

【００６４】

図３乃至図５で参照されるように、ピア-ベクトル・マシン４０の動作（特にホストプロセッサ４２、パイプライン回路８０、並びに、ファームウェア・メモリ５２の動作）は、本発明の実施例に従って以下に議論される。

【００６５】

ピア-ベクトル・マシン４０は先ず電源投入されると、ホストプロセッサ４２は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に議論されているようにそれ自体を初期化し、加速器４４はそれ自体を部分的に初期化する。より詳細には、この部分的初期化中、パイプライン回路８０はメモリ５２の区分１１４から初期コンフィギュレーション・ファームウェアをダウンロードする。先に議論されたように、初期コンフィギュレーションにおいて、パイプライン回路８０は少なくとも通信インターフェース８２及び試験回路（不図示）を含む。パイプライン回路８０がその初期コンフィギュレーションで構成された後、試験回路はパイプライン回路及びデータ・メモリ８１の自己試験を実行し、その自己試験の結果を試験ポート９６及び試験バス６３を介してホストプロセッサ４２に提供する。ファームウェア・メモリ５２も自己試験を実行し得て、図５と連携して先に議論されたように、その結果を試験ポート１０４及び試験バス６３を介してホストプロセッサ４２に提供する。

【００６６】

次に、ホストプロセッサ４２は加速器４４の部分的初期化中に例外が生じたかを決定する。例えば、ホストプロセッサ４２は試験バス６３からの自己試験結果を分析して、パイプライン回路８０、データ・メモリ８１、並びに、ファームウェア・メモリ５２が適切に機能しているかを決定する。

【００６７】

もし例外が生じたならば、ホストプロセッサ４２はそれを所定方式で取り扱う。例えば、もしホストプロセッサ４２がパイプライン回路８０から自己試験結果を受信しなければ、試験バス６３を介して、初期コンフィギュレーション・ファームウェアがファームウェア・メモリ５２の区分１１４内に記憶されているかを検査し得る。もしその初期コンフィギュレーション・ファームウェアが記憶されていなければ、ホストプロセッサ４２はその初期コンフィギュレーション・ファームウェアをパイプライン・バス５０或は試験バス６

３を介して区分１１４にロードして、パイプライン回路８０にこのファームウェアをダウンロードさせてから、自己試験の結果を分析し得る。例外のホストプロセッサの取り扱いは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に更に議論されている。

【００６８】

もし例外が生じなければ、ホストプロセッサ４２はパイプライン・ユニット７８からプロファイル識別子を読み取り、引き続き加速器コンフィギュレーション・レジストリ７０からパイプライン・ユニットの対応するプロファイルを獲得する。ファームウェア・メモリ５２の区分１２０からの代わりに、レジストリ７０からプロファイルを獲得することはしばしばより好ましく、その理由は、もしパイプライン回路８０がＡＳＩＣであれば、パイプライン・ユニット７８がファームウェア・メモリ等の不揮発性メモリを含まないことがあり得るからである。もしそのプロファイル識別子がパイプライン回路８０がＡＳＩＣであることを示せば、ホストプロセッサ４２はファームウェアがパイプライン回路にダウンロードされる必要性が何等ないことを決定する。代替的には、ホストプロセッサ４２（図３）はファームウェア・メモリ５２の区分１２０からプロファイルを獲得し得る。この代替例において、プロファイル識別子を記憶することはパイプライン・ユニット７８にとって不必要であるが、該パイプライン・ユニットはそのプロファイルが区分１２０から故意ではなく削除される場合にプロファイル識別子を記憶し得る。

【００６９】

次に、パイプライン・ユニット７８（図４に１つのみ示されている）の全てからプロファイル識別子を読み取った後、ホストプロセッサ４２は加速器４４におけるパイプライン・ユニット７８全てのマップを効果的に生成し、そのマップを例えばハンドラー・メモリ６８内に記憶する。

【００７０】

次いで、各パイプライン・ユニット７８に対して、ホストプロセッサ４２はプロファイルからパイプライン回路８０の所望の動作コンフィギュレーションの識別子を抽出する。加速器４４の初期化中に所望の動作コンフィギュレーションを抽出することは、初期化に先行してそのプロファイルを単に更新するによってパイプライン回路８０の動作を変更させることを可能とする。

【００７１】

次に、ホストプロセッサ４２は所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ５２に記憶されているかを決定する。例えば、ホストプロセッサ４２は、プログラミング・バス１１０及び（パイプライン回路８０が初期コンフィギュレーションにあるので、通信インターフェースが存在するので）通信インターフェース８２を介してメモリ区分１１８からそのコンフィギュレーション記述を読み取ることができ、所望のファームウェアが区分１１６$_1$－１１６$_i$の何れかに記憶されているかを決定する。代替的には、ホストプロセッサ４２は試験バス６３及び試験ポート１０４を介してメモリ５２から直にコンフィギュレーション記述を読み取り得る。

【００７２】

もし所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ５２に記憶されていなければ、ホストプロセッサ４２は通信インターフェース８２、プログラミング・ポート９４，１０６，並びに、プログラミング・バス１１０を介して加速器コンフィギュレーション・レジストリ７０からファームウェア・メモリの区分１１６$_1$－１１６$_i$の内の１つにこのファームウェアをロードする。もしファームウェアがレジストリ７０内になければ、ホストプロセッサ４２は外部ライブラリ（不図示）からファームウェアを検索し得るか、或は、例外指標を生成し得て、システム動作（不図示）がそのファームウェアをレジストリ７０にロードし得るように為す。

【００７３】

次に、ホストプロセッサ４２は、パイプライン回路８０にポート１０８、コンフィギュレーション・バス１１２、並びに、ポート９８を介してメモリ５２の対応する区分１１６

,－１１６,から所望のファームウェアをダウンロードさせるように命令する。

【００７４】

パイプライン回路８０が所望のファームウェアをダウンロードした後、所望の動作コンフィギュレーションとなって、データ処理を始める準備が為される。しかし、パイプライン回路８０がその所望動作コンフィギュレーションとなった後でさえ、ホストプロセッサ４２は通信インターフェース８２或は試験バス６３を介して新しいファームウェアをメモリ５２の区分１１６,－１１６,にロードし得る。例えば、新しいファームウェアをロードすべく、ホストプロセッサ４２は先ずパイプライン回路８０にメモリ５２の区分１１４からファームウェアを再ロードさせ得て、パイプライン回路が再び初期コンフィギュレーションとなるように為す。次いで、ホストプロセッサ４２はパイプライン・バス５０及び通信インターフェース８２を介して新しいファームウェアを区分１１６,－１１６,の内の１つにロードする。次に、ホストプロセッサ４２はパイプライン回路８０にその新しいファームウェアをダウンロードさせて、パイプライン回路がその新しい動作コンフィギュレーションとなるように為す。初期コンフィギュレーションであるときにのみ、パイプライン回路８０に新しいファームウェアをメモリ５２にロードさせることは、２つの長所をもたらす。第１として、それはパイプライン回路８０が動作コンフィギュレーションにある際に該パイプライン回路がメモリ５２に記憶されているファームウェアを故意ではなく改変することを防止する。第２として、それは動作コンフィギュレーションに、さもなければファームウェアをメモリ５２に書き込むために必要とされる回路用に使用されることになるパイプライン回路８０のリソースを利用させることを可能とする。

【００７５】

図６は、本発明の別の実施例に従った図３のパイプライン加速器４４のパイプライン・ユニット１２４のブロック線図である。

【００７６】

パイプライン・ユニット１２４は、そのパイプライン・ユニット１２４が多数のパイプライン回路８０（ここでは２つのパイプライン回路８０ａ及び８０ｂ）と、各パイプライン回路に対して１つずつのメモリである多数のファームウェア・メモリ（ここでは２つのメモリ５２ａ及び５２ｂ）と、をきむことを除いて、図４のパイプライン・ユニット７８と類似している。パイプライン回路８０ａ及び８０ｂとファームウェア・メモリ５２ａ及び５２ｂとの組み合わせはプログラマブル回路ユニットを形成する。１つの具現化例において、メモリ５２ａ及び５２ｂの各々は、ファームウェア５２ｂが、パイプライン・ユニット１２４のプロファイルを記憶し、さもなければメモリ５２ａの区分１２０と重複するような区分１２０を省略し得ることを除いては、図５に示されるように仕切られる。代替的には、パイプライン回路８０ａ及び８０ｂはメモリ５２ａ及び５２ｂと動作的に類似している各区分を含む単一ファームウェア・メモリを共有し得る。パイプライン回路８０の数を増大することは、典型的には、ハードウェアに組み込まれたパイプライン７４,－７４,の数ｎに関する増大を許容し、よってパイプライン・ユニット７８と比較してのパイプライン・ユニット１２４の機能に関する増大を許容する。更には、パイプライン回路８０ａ及び８０ｂの何れか一方或は双方はＡＳＩＣであり得て、その場合、対応するファームウェア・メモリ（又は対応する複数のファームウェア・メモリ）５２は省略し得る。

【００７７】

パイプライン・ユニット１２４の構造及び動作の更なる詳細は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献４に議論されている。

【００７８】

パイプライン回路８０ａは試験ポート９６ａ及びハード・コンフィギュレーション・ポート９８ａを含み、それらが図４の試験ポート９６及びハード・コンフィギュレーション・ポート９８とそれぞれ類似している。そして図４のパイプライン回路８０と同じように、パイプライン回路８０ａはプログラミング・ポート９４を有する通信インターフェース８２を含む。

【００７９】
パイプライン回路８０ｂは試験ポート９６ａ及びハード・コンフィギュレーション・ポート９８ｂを含み、それらも図４の試験ポート９６及びハード・コンフィギュレーション・ポート９８とそれぞれ類似している。そして、ホストプロセッサ４２（図３）が以下に議論されるようにパイプライン回路８０ａの通信インターフェース８２を介してファームウェア・メモリ５２ｂをプログラムできるので、パイプライン回路８０ｂはプログラミング・ポートを含まない。
【００８０】
ファームウェア・メモリ５２ａは試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート１０４ａ，１０６ａ，１０８ａを含み、それらが図４の試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート１０４，１０６，１０８とそれぞれ類似している。試験ポート１０４ａは試験バス６３と結合され、プログラミング・ポート１０６ａはプログラミング・バス１１０を介して通信インターフェース８２のプログラミング・ポート９４ａと結合され、そしてハード・コンフィギュレーション・ポート１０８ａはコンフィギュレーション・バス１１２ａを介してパイプライン回路８０ａのハード・コンフィギュレーション・ポート９８ａと結合されている。
【００８１】
同様に、ファームウェア・メモリ５２ｂは試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート１０４ｂ，１０６ｂ，１０８ｂを含み、それらが図４の試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート１０４，１０６，１０８とそれぞれ類似している。試験ポート１０４ｂは試験バス６３と結合され、プログラミング・ポート１０６ｂはプログラミング・バス１１０を介して通信インターフェース８２のプログラミング・ポート９４ａと結合され、そしてハード・コンフィギュレーション・ポート１０８ｂはコンフィギュレーション・バス１１２ｂを介してパイプライン回路８０ｂのハード・コンフィギュレーション・ポート９８ｂと結合されている。
【００８２】
図３、図５、並びに、図６で参照されるように、ピア・ベクトル・マシン４０の動作（特にホストプロセッサ４２、パイプライン回路８０ａ及び８０ｂ、ファームウェア・メモリ５２ａ及び５２ｂの動作）は本発明の実施例に従って以下に議論されている。
【００８３】
ピア・ベクトル・マシン４０が先ず電源投入されると、ホストプロセッサ４２は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３で議論されたようにそれ自体を初期化し、加速器４４はそれ自体を部分的に初期化する。より詳細には、この部分的初期化中、パイプライン回路８０ａ及び８０ｂは、それぞれ、ファームウェア・メモリ５２ａ及び５２ｂの区分１１４ａ及び１１４ｂから初期コンフィギュレーション・ファームウェアをダウンロードする。それぞれの初期コンフィギュレーションにおいて、パイプライン回路８０ａは少なくとも通信インターフェース８２及び試験回路（不図示）を含み、パイプライン回路８０ｂは少なくとも試験回路（不図示）を含む。パイプライン回路８０ａ及び８０ｂがそれらの各初期コンフィギュレーションに構成された後、各パイプライン回路内の試験回路はパイプライン回路の各自己試験を実行し（パイプライン回路８０ａ及び８０ｂの一方或は双方の試験回路もデータ・メモリ８１を試験し得て）試験ポート９６ａ及び９６ｂのそれぞれと試験バス６３とを介してホストプロセッサ４２にそれら自己試験の結果を提供する。ファームウェア・メモリ５２ａ及び５２ｂも各自己試験を実行し得て、図５と連携されて先に議論されたように、試験ポート１０４ａ及び１０４ｂのそれぞれと試験バス６３とを介してホストプロセッサ４２にその結果を提供する。
【００８４】
次に、ホストプロセッサ４２は加速器４４の部分的初期化中に例外が生じたかを決定する。例えば、ホストプロセッサ４２は試験バス６３からの自己試験結果を分析して、パイプライン回路８０ａ及び８０ｂ、データ・メモリ８１、並びに、ファームウェア・メモリ

５２ａ及び５２ｂが適切に機能しているかを決定する。

【００８５】

もし例外が生じたならば、ホストプロセッサ４２はそれを所定の方式で取り扱う。例えば、もしホストプロセッサ４２がパイプライン回路８０ａから自己試験結果を受信しなければ、初期コンフィギュレーション・ファームウェアがファームウェア・メモリ５２ａの区分１１４ａに記憶されているかを試験バス６３を介して検査し得る。もし初期コンフィギュレーション・ファームウェアが記憶されていなければ、ホストプロセッサ４２はその初期コンフィギュレーション・ファームウェアを区分１１４ａにロードして、パイプライン回路８０ａにそのファームウェアをダウンロードさせてから、自己試験の結果を分析し得る。この例はパイプライン回路５０ｂ及びファームウェア・メモリ５２ｂにも適用される。例外のホストプロセッサの取り扱いは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献３に更に議論されている。

【００８６】

もし例外が何等生じなかったならば、ホストプロセッサ４２はパイプライン・ユニット１２４からプロファイル識別子を読み取り、続いて、図４と連携して先に議論されたように、加速器コンフィギュレーション・レジストリ７０から或はファームウェア・メモリ５２ａの区分１２０からパイプライン・ユニットの対応するプロファイルを獲得する。

【００８７】

次に、パイプライン・ユニット１２４（図４には１つのみ示されている）の全てからプロファイル識別子を読み取った後、ホストプロセッサ４２は加速器４４における全パイプライン・ユニットのマップを効果的に生成して、そのマップを例えばハンドラー・メモリ６８に記憶する。

【００８８】

次いで、ホストプロセッサ４２はプロファイルからパイプライン回路８０ａ及び８０ｂの所望の動作コンフィギュレーションの識別子を抽出する。加速器４４の初期化中に所望の動作コンフィギュレーションを抽出することは、そのプロファイルを初期化に先行して単に更新することによってパイプライン回路８０ａ及び／或は８０ｂの動作を変更させることを可能とする。

【００８９】

次に、ホストプロセッサ４２は所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ５２ａ及び５２ｂに記憶されているかを決定する。例えば、ホストプロセッサ４２はプログラミング・バス１１０及び（パイプライン回路８０ａが初期コンフィギュレーションであるので、通信インターフェースが存在するので）通信インターフェース８２を介してメモリ５２ａのメモリ区分１１８ａからコンフィギュレーション記述を読み取ることができ、所望のファームウェアが区分１１６ａ$_1$－１１６ａ$_i$の何れかに記憶されていることを決定する。代替的には、ホストプロセッサ４２は試験バス６３及び試験ポート１０４ａを介してメモリ５２ａから直にコンフィギュレーション記述を読み取り得る。この例もパイプライン回路５０ｂ及びファームウェア・メモリ５２ｂに適用される。

【００９０】

もし所望の動作コンフィギュレーションの一方或は双方を表すファームウェアがファームウェア・メモリ５２ａ及び／或は５２ｂに記憶されていなければ、ホストプロセッサ４２は加速器コンフィギュレーション・レジストリ７０からそのファームウェアを通信インターフェース８２、プログラミング・ポート９４、１０６、並びに、プログラミング・バス１１０を介して適切なファームウェア・メモリの区分１１６$_1$－１１６$_i$の内の１つにロードする。例えば、もしパイプライン回路８０ｂの所望の動作コンフィギュレーションを表すファームウェアがメモリ５２ｂに記憶されていなければ、ホストプロセッサ４２はそのファームウェアをレジストリ７０から、インターフェース８２、プログラミング・ポート９４、１０６ｂ、並びに、プログラミング・バス１１０を介して区分１１６ｂ$_1$－１１

６ｂｉの内の１つにロードする。もしファームウェアがレジストリ７０になければ、ホストプロセッサ４２はそのファームウェアを外部ライブラリ（不図示）から検索し得るか、或は、例外指標を生成し得て、システムオペレータ（不図示）がそのファームウェアをレジストリ７０にロードし得るように為す。

【００９１】

次にホストプロセッサ４２は、パイプライン回路８０ａに、ポート１０８ａ、コンフィギュレーション・バス１１２ａ、並びに、ポート９８ａを介してメモリ５２ａの対応する区分１１６ａ１－１１６ａｉから所望のファームウェアをダウンロードするように命令し、パイプライン回路８０ｂに、ポート１０８ｂ、コンフィギュレーション・バス１１２ｂ、並びに、ポート９８ｂを介してメモリ５２ｂの対応する区分１１６ｂ１－１１６ｂｉから所望のファームウェアをダウンロードするように命令する。

【００９２】

パイプライン回路８０ａ及び８０ｂが所望のファームウェアをダウンロードした後、それは所望の動作コンフィギュレーションにあって、データ処理を始める準備が為される。しかし、パイプライン回路８０ａ及び８０ｂがそれらの所望の動作コンフィギュレーションになった後でさえ、ホストプロセッサ４２は新しいファームウェアを通信インターフェース８２或は試験バス６３を介して、図４と連携されて先に議論されたものと同様な方式でメモリ５２ａ及び５２ｂの区分１１６１－１１６ｉにロードし得る。

【００９３】

先行する議論は当業者が本発明を作製し使用することを可能とすべく提示されている。種々実施例への様々な変更は当業者には容易に明かであろうし、ここでの包括的な原則は本発明の精神及び範囲から逸脱することなしに他の実施例及び適用例に適用され得る。よって、本発明は図示された実施例に限定されることが意図されておらず、ここに開示された原理及び特徴と一貫した最も広い範囲と一致されるべきものである。

【図面の簡単な説明】

【００９４】

【図１】図１は、従来の多数プロセッサ・アーキテクチャを有する計算マシンのブロック線図である。

【図２】図２は、従来のハードウェアに組み込まれたパイプラインのブロック線図である。

【図３】図３は、本発明の実施例に従ったピア-ベクトル・アーキテクチャを有する計算マシンのブロック線図である。

【図４】図４は、本発明の実施例に従った図３のパイプライン加速器におけるパイプライン・ユニットのブロック線図である。

【図５】図５は、本発明の実施例に従った図４のファームウェア・メモリの論理区画の線図である。

【図６】図６は、本発明の別の実施例に従った図３のパイプライン加速器のパイプライン・ユニットのブロック線図である。

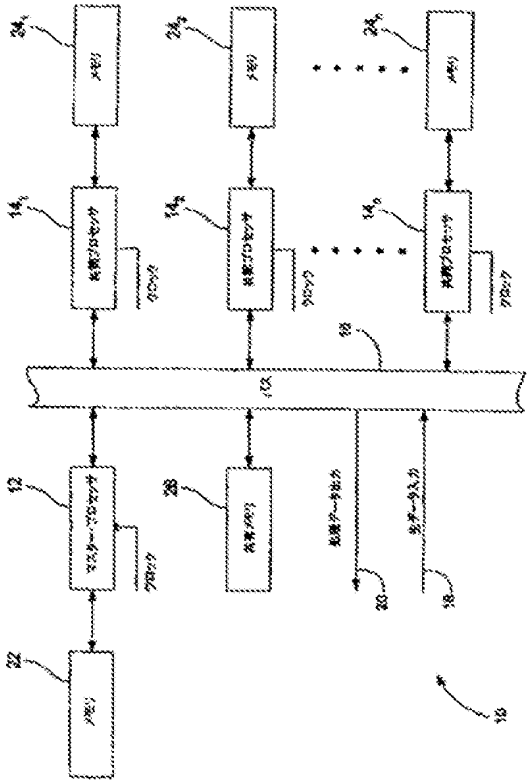
【符号の説明】

【００９５】

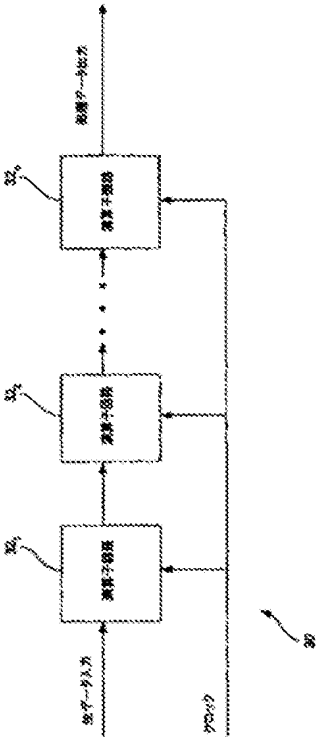
１０　計算マシン
１４　共同プロセッサ
４０　ピア-ベクトル・マシン
４２　ホストプロセッサ
４４　パイプライン加速器
４６　プロセッサ・メモリ
４８　インターフェース・メモリ
５０　パイプライン・バス
５２　ファームウェア・メモリ
５４　生データ入力ポート

５８　処理済みデータ出力ポート
６１　ルータ
６２　処理ユニット
６６　処理ユニット・メモリ
６８　ハンドラー・メモリ
７０　加速器コンフィギュレーション・レジストリ
７２　メッセージ・コンフィギュレーション・レジストリ
７４　ハードウェアに組み込まれたパイプライン
７８　パイプライン・ユニット
８０　パイプライン回路
８６　パイプライン・コントローラ
８８　例外マネージャ
９０　コンフィギュレーション・マネージャ
９１　工業規格バス・インターフェース
９３　通信バス

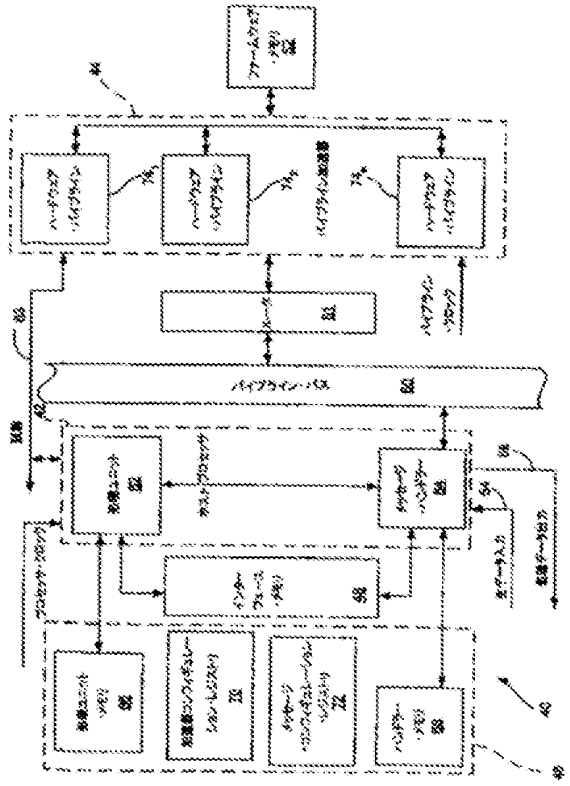
【図１】



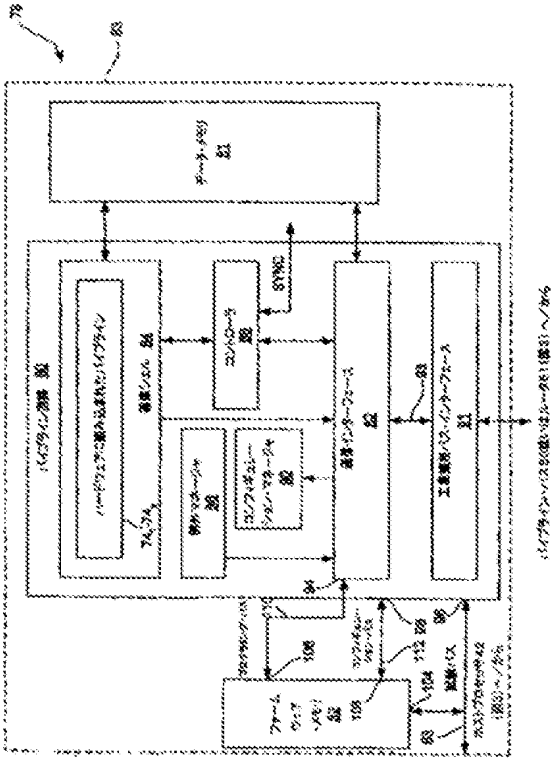
【図２】



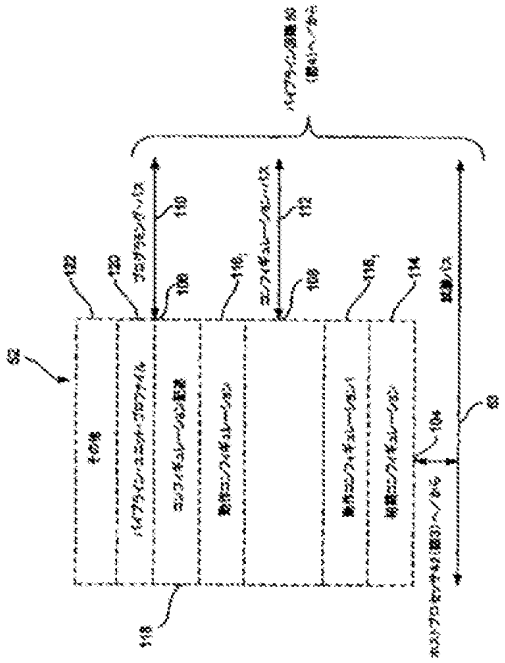
【図３】



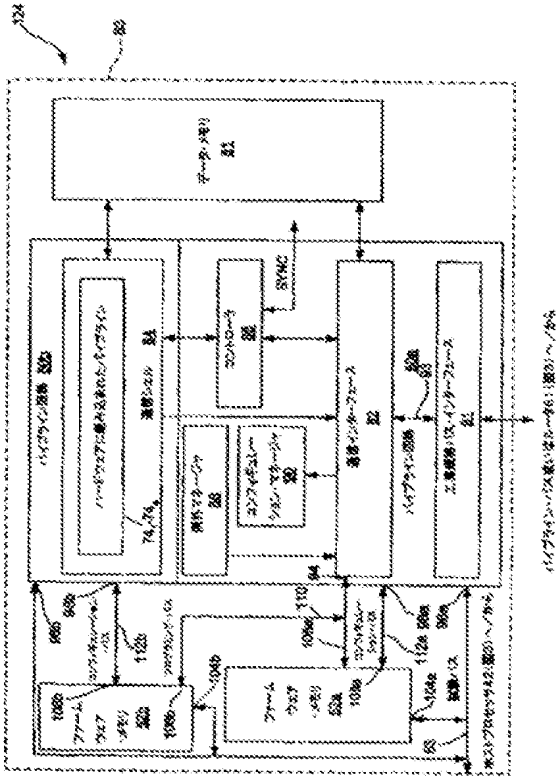
【図４】



【図５】



【図６】



フロントページの続き

(31)優先権主張番号 10/684,053
(32)優先日 平成15年10月9日(2003.10.9)
(33)優先権主張国 米国(US)
(31)優先権主張番号 10/684,057
(32)優先日 平成15年10月9日(2003.10.9)
(33)優先権主張国 米国(US)
(31)優先権主張番号 10/684,102
(32)優先日 平成15年10月9日(2003.10.9)
(33)優先権主張国 米国(US)

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

(74)代理人 100135585
弁理士 内尾 務
(72)発明者 ラーブ，ジョン，ダブリュ．
アメリカ合衆国 バージニア州 ２０１１０ マナサッス，リバー クレスト ロード ９３５０
(72)発明者 ジャクソン，ラリー
アメリカ合衆国 バージニア州 ２０１１０ マナサッス，リバー クレスト ロード ９３５０
(72)発明者 ジョウンズ，マーク
アメリカ合衆国 バージニア州 ２０１２０ セントレビル，オークマー プレイス １５３４２
(72)発明者 カーサロ，トロイ
アメリカ合衆国 バージニア州 ２２７０１ カルペパー，ケストラル コート １５２４
Fターム(参考) 5B076 BB06 EB01
5B176 BB06 EB01